**Exercise 7: Financial Forecasting**

**1.Recursive Algorithms**

**Recursion** is a method where the solution to a problem depends on solutions to smaller instances of the same problem. A recursive function calls itself with a modified argument, aiming to reach a base case where the function does not call itself.

**Key Components of Recursion:**

1. **Base Case**: The condition under which the recursive function stops calling itself. Without a base case, the function would call itself indefinitely, leading to a stack overflow.
2. **Recursive Case**: The part of the function where it calls itself with a modified argument, moving towards the base case.

**Example:**

To calculate the factorial of a number n:

factorial(n)=n× factorial(n−1)

**How Recursion Simplifies Problems**

Recursion can simplify problems that have a natural recursive structure, such as those involving:

* **Divide and Conquer**: Breaking a problem into smaller sub-problems of the same type.
* **Self-Similar Structures**: Problems where the solution involves solving the same problem on smaller scales.

**Analysis of the Recursive Algorithm**

**Time Complexity**

The given recursive method calculateFutureValue computes the future value of an investment using the following recurrence relation:

FV(P,r,n)=FV(P×(1+r),r,n−1)

Here:

* PPP is the principal (initial amount).
* rrr is the growth rate.
* nnn is the number of years.

For each year, the method calls itself once with n−1 until it reaches the base case where n=0. Therefore, the time complexity is:

T(n)=T(n−1) + O(1)

This gives us a linear time complexity:

T(n)=O(n)

**Optimizing the Recursive Solution**

While the current recursive solution is efficient enough for a small number of years, it can be optimized further by avoiding redundant calculations. Here are a couple of optimization strategies:

1. **Memoization**: Store previously computed results to avoid recalculating them.
2. **Iterative Approach**: Convert the recursive solution to an iterative one, which is generally more efficient in terms of both time and space complexity.

**Time Complexity**: The recursive solution has a time complexity of O(n)O(n)O(n), where nnn is the number of years.

**Memoization**: Using memoization can optimize the recursive solution by storing and reusing previously computed results.

**Iterative Approach**: Converting the recursive solution to an iterative one provides a more efficient and straightforward solution, avoiding the overhead of recursive calls and potential stack overflow issues.